UNIVERSIDAD AUTONÓMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DETECCIÓN DE OBJETOS EN IMÁGENES URBANAS DE GOOGLE STREET VIEW

Autor: Paula Guerra Toni
Tutor: Pablo Carballeira López
Ponente: José María Martínez Sánchez

Julio 2019

DETECCIÓN DE OBJETOS EN IMÁGENES URBANAS DE GOOGLE STREET VIEW

AUTOR: Paula Guerra Toni

TUTOR: Pablo Carballeira López

PONENTE: José María Martínez Sánchez

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2019

Trabajo parcialmente financiado por el Gobierno de España bajo el proyecto TEC2017-88169-R (MobiNetVideo)

Resumen

En este Trabajo Fin de Grado se presentan como objetivos el desarrollo de una herramienta para la formación de una base de datos mediante la extracción de imágenes de Google Street View a lo largo de distintas rutas y la prueba y evaluación de un algoritmo de detección de objetos sobre estas imágenes.

En primer lugar, se desarrolló una herramienta que extrae imágenes a lo largo de rutas seleccionadas. Estas rutas son las rutas óptimas calculadas por los algoritmos de Google Maps, dados un origen, un destino y el modo de transporte a utilizar. Las imágenes extraídas no son panorámicas, pero permiten la variación de algunos parámetros como son el ángulo de toma de la imagen, tanto horizontal (heading) como vertical (pitch), y el campo de visión (fov), para poder representar todo el entorno del punto de toma. Así, se recogen imágenes con distintos objetos, características y distorsiones.

En segundo lugar, una vez creada la base de datos con nuestras imágenes, se aplica sobre ella el modelo de detección de objetos de YOLOv3, previamente entrenado mediante la base de datos de COCO, para detectar determinados objetos en ellas. A partir de estos resultados se crea un algoritmo para la representación óptima de estas detecciones sobre las respectivas imágenes.

Por último, para completar el objetivo, se implementa un sistema para evaluar la calidad de las detecciones sobre las imágenes, mediante el cálculo y la representación de la relación entre la precision y la sensibilidad del modelo. Con este fin, se crea un Ground Truth sobre algunas imágenes y un programa que lo compara con las detecciones producidas por el detector sobre esas imágenes. Además, se estudian y evalúan algunos casos en los que la detección no es tan eficiente debido a que algunas imágenes contienen áreas y objetos distorsionados.

Palabras clave

Bases de datos, Google Street View, Google Maps, imágenes, entornos urbanos, rutas, detección de objetos, YOLO, YOLOv3 y COCO.

Abstract

In this Bachelor Thesis proposes two objectives: the development of an algorithm for the creation of a data base formed through the extraction of images from Google Street View along different routes, and testing and evaluating an object detector over these images.

In the first place, a tool which extract images along selected routes is developed. These routes are the optimal ones, which are calculated trough the Google Maps algorithms, from an origin, a destiny and the transport mode to be used. The extracted images are not panoramic ones, but they allow a variation of parameters. Some of these parameters are the angle from which the images are taken, horizontal (heading) as well as vertical (pitch), and the filed of view (fov), to be able to represent the whole surroundings where the images are taken. In this way, the images with different objects, characteristics and distortions are collected.

In the second place, once the data base with our images is created, the object detection model of YOLOv3, pretrained with the data base COCO, is applied to it, in order to detect certain objects in them. From these results an algorithm is created to the optimal representation of these detections on the respective images.

Finally, in order to complete the objective, we implemented a system to evaluate the quality of the detection on the images, through the calculation and representation of the relation between the precision and the recall of the model. To this aim, we create a Ground Truth over some images as well as a program which compares it with the detections produced by the detector over these images. Besides, some cases in which the detector was not efficient enough, because of some images have distorted areas and objects, are studied and evaluated.

Keywords

Databases, Google Street View, Google Maps, images, urban environments, routes and object detection.

Agradecimientos

En primer lugar, me gustaría dar las gracias a mi tutor, Pablo Carballeira López, por toda su ayuda en la realización de este Trabajo de Fin de Grado.

También gustaría dar las gracias a mi familia, que siempre ha creído en mí y me ha apoyado en todo momento.

Por último, agradecer a mis compañeros, con los que he compartido tantos momentos. Sin ellos no habría sido lo mismo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

En los últimos años ha habido un gran avance en los algoritmos de visión artificial, siendo una de las tecnologías más relevantes de nuestro futuro. En la actualidad ya es utilizada en muchos de los campos de nuestra vida cotidiana como en la seguridad, la salud, el entretenimiento, la agricultura, la industria, etc. Uno de los campos más significativos en los que se está incorporando la visión artificial es en el transporte, y con este, la conducción autónoma y la navegación sin uso de mapas, sino solo con imágenes en la ruta. Aunque la navegación con imágenes está avanzando y desarrollándose, entre otros, por los trabajos y aportes de la empresa DeepMind, es un campo en el cual queda aún mucho trabajo por realizar hasta alcanzar la madurez [1]. La visión artificial emplea imágenes y, gracias a redes neuronales entrenadas, trata la información para realizar el proceso correspondiente. [2][3]

Para entrenar y probar las redes neuronales utilizadas en visión artificial se utilizan bases de datos masivas. Por esto, es importante la existencia de estas grandes bases de datos variadas para todos los campos en los que se desarrollen estas tecnologías. Uno de estos campos, como ya se ha explicado anteriormente, es el transporte. Para este, Google Street View cuenta con una base de datos muy amplia compuesta por imágenes urbanas de carreteras alrededor del mundo. Sin embargo, se encuentra una escasa cantidad de programas de extracción de imágenes de Google Street View y, sobre todo, que estos extraigan bases de datos amplias de forma automática y estructurada.

La motivación de este trabajo es lograr la creación de una base de datos a partir de imágenes de Google Street View. Estas imágenes serían extraídas con distintas características y alteraciones geométricas de los objetos representados en ellas. Una vez obtenida esta base de datos, se evaluará sobre las imágenes algoritmos actuales de detección de objetos, para valorar las variaciones de las características y deformaciones en dichos objetos.

1.2 Objetivos

Uno de los principales objetivos de este Trabajo de Fin de Grado consiste en crear una herramienta con la que se puedan extraer las imágenes de la base de datos de Google Street View a lo largo de unas rutas. Con el conjunto de imágenes obtenidas de las rutas se creará una propia base de datos.

Las imágenes recogidas por Google Street View son imágenes que cubren todo el campo de visión abarcado por las cámaras en cada punto de captura. Las cámaras de Google toman un conjunto de imágenes con ángulos diferentes, de manera que se represente todo el campo de visión posible. Según los diferentes ángulos, los objetos representados en una imagen se verán con un nivel de distorsión geométrica distinta, siendo los más cercanos a

la cámara los más distorsionados. Se propone la extracción de imágenes con un nivel de distorsión geométrico alto en sus objetos.

El otro objetivo principal es la prueba de un detector de objetos de alto nivel sobre nuestras imágenes extraídas de Google Street View. Este detector, previamente entrenado, se probará sobre las distintas rutas con características y objetos distintos cada una. Por último, se evaluará el detector sobre algunas clases de objetos.

1.3 Organización de la memoria

La memoria de este trabajo de fin de grado se organiza en los siguientes capítulos:

- **Capítulo 1. Introducción:** Motivación y objetivos del trabajo.
- **Capítulo 2. Estado del arte:** Estudio de las tecnologías de creación de bases de datos de imágenes de Google Street View y de las tecnologías de detección de objetos en imágenes urbanas.
- **Capítulo 3. Diseño y desarrollo:** Descripción de la tecnología desarrollada para la creación de una base de datos de imágenes de Google Street View y detalles sobre la detección YOLO de objetos sobre estas imágenes.
- **Capítulo 4. Integración, pruebas y resultados:** Evaluación del sistema mediante resultados del modelo y un algoritmo.
- **Capítulo 5.** Conclusiones y trabajo futuro.
- **Referencias**
- **Glosario**
- **Anexos**

2 Estado del arte

El objetivo de este capítulo es exponer y desarrollar las herramientas existentes para generar una base de datos compuesta por imágenes de Google Street View (Sección 2.1) y los distintos algoritmos de detección de objetos (Sección 2.2).

2.1 Bases de datos a partir de imágenes de Google Street View

Desde 2007 Google cuenta con la plataforma de Google Street View, como complemento de Google Maps y Google Earth. Este servicio proporciona millones de imágenes panorámicas a nivel de calle para representar virtualmente nuestro entorno. Estas imágenes son imágenes panorámicas en localizaciones discretas en el espacio, las cuales forman su base de datos. Actualmente este servicio está presente en más de 130 países a lo largo del planeta, como se muestra en la Figura 2.1. Para ello Google cuenta con una flota de equipos y recursos (como son los vehículos de Street View, mochilas especiales, carritos, triciclos e incluso motos de nieve) dependiendo su utilización del terreno. Todos estos cuentan con una cámara especial 360º. Estas imágenes panorámicas son tomadas por cámaras de alta definición que proporcionan una cobertura de 360º horizontales y 180º verticales, con un campo de visión (Field of view) máximo [4].

Para poder extraer estas imágenes de su base de datos, Google Maps dispone públicamente, pero con restricciones, una API llamada Street View Static API, desarrollada en la siguiente Sección 2.1.1.

Figura 2.1: Cobertura de Google Street View. En azul las zonas donde Google Street View está disponible.

Google Maps, por su parte, nos permite crear rutas desde un punto A hacia un punto B. Este servicio nos devuelve la ruta más eficiente entre todas las posibles. El cálculo de rutas posibles depende del modo de transporte, de si hay algún punto especificado de parada entre origen y destino, o de las opciones de evitar peajes, transbordadores y autopistas principales entre otros. El tiempo de viaje es el factor principal que se optimiza, pero otros factores como la distancia, el número de giros y muchos más, pueden ser tomados en cuenta para elegir la ruta óptima. La Figura 2.2 muestra un ejemplo de ruta óptima.

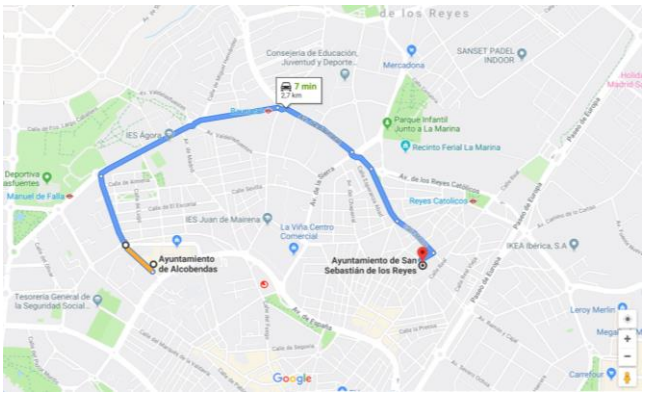

Figura 2.2: Ruta óptima propuesta por Google Maps desde un punto A (Ayuntamiento de Alcobendas) hacia un punto B (Ayuntamiento de San Sebastián de los Reyes).

Para poder calcular estas rutas Google Maps dispone públicamente de una API llamada Directions API, desarrollada en la siguiente Sección 2.1.1.

2.1.1 APIs de Google

Google Maps ofrece una serie de funciones y especificaciones desarrolladas en unas APIs. El procedimiento de extracción de imágenes es guiado por las siguientes APIs:

- STREET VIEW STATIC API [5]
  Biblioteca de funciones, procedimientos y subrutinas que permite extraer imágenes de la base de datos de Google Street View. Las imágenes panorámicas, como la de la Figura 2.3, generalmente se obtienen tomando varias fotos desde una posición y uniéndolas usando un software panorámico (stitching). El panorama de 360 grados resultante es la proyección en la superficie bidimensional de la imagen esférica, representada en la Figura 2.4.

Figura 2.3: Imagen panorámica.

Figura 2.4: Esfera de una imagen 360º.

La solicitud de una imagen extrae una subimagen no panorámica de las que está formada la imagen panorámica de 360 grados. Esta subimagen se especifica mediante unos parámetros. Sobre la esfera representada en la Figura 2.4, las posibles subimagenes extraídas serían las distintas imágenes adyacentes.

La solicitud se define por una URL HTTP estándar formada por unos parámetros. Estos parámetros pueden ser obligatorios u opcionales. A continuación, se describen algunos de ellos:

- Ubicación (Parámetro obligatorio):

  Ya sea

  - Location: (Parámetro obligatorio) Cadena de texto o un valor de latitud y longitud. La imagen de salida será del lugar más cercano a esta ubicación. La base de Google Street View se actualiza regularmente y las fotografías pueden ser tomadas desde puntos ligeramente distintos, por lo que es posible que la imagen sea tomada en una ubicación diferente.

  ó

  - Pano: ID de panorama específico. Este parámetro es único para cada imagen panorama.

- Size (Parámetro obligatorio): Resolución en píxeles de la imagen de salida.

- Key (Parámetro obligatorio): Clave API para la utilización de funciones que disponen las APIs de Google. Esta clave identifica su aplicación para fines de administración de cuotas en Google Cloud Platform. Su uso para esta API está limitado a 500 consultas por segundo. Por cada consulta, al usuario se le carga una cuota. La tabla de precios se puede consultar en [6]. Google Cloud cuenta con un plan Premium y ofrece una prueba gratuita que consiste en un año o 300 USD gratis.

- Signature (Parámetro opcional): Firma digital para verificar la autorización del uso de la clave API.

- Fov (Field of view) (Parámetro opcional): Campo de visión horizontal de la imagen. Se expresa en grados, con un valor máximo permitido de 120 siendo este el mayor campo de visión (menor zoom). El valor por defecto es de 90.
  Las Figura 2.5 muestra un ejemplo de 4 imágenes recogidas del mismo punto en una ruta, pero variando el campo de visión.

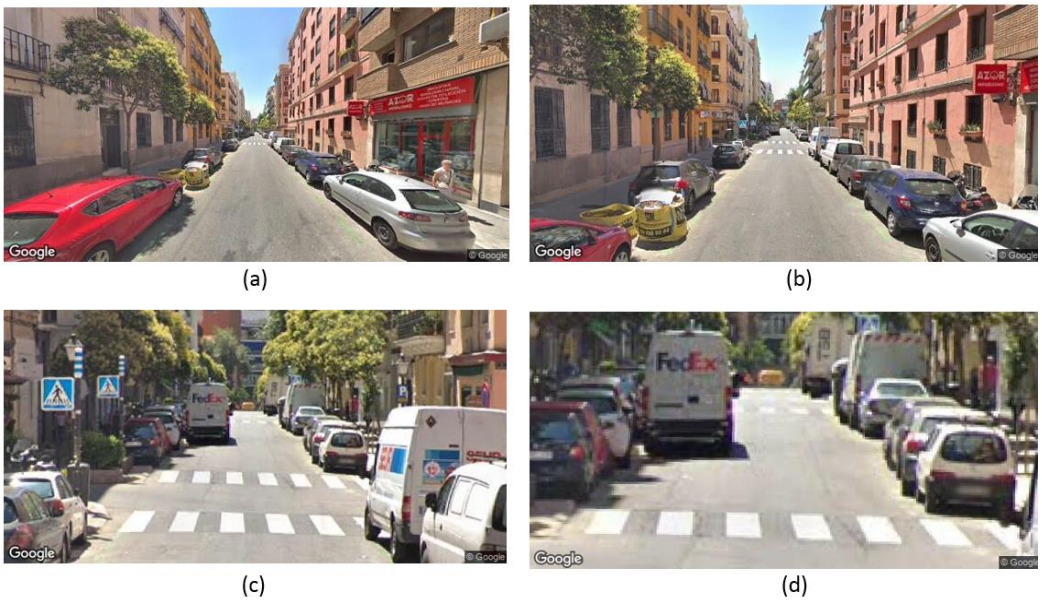

Figura 2.5: Ejemplo de variación del valor del fov en el mismo punto de toma de imágenes donde (a) es una imagen con fov 120, pitch 0 y heading por defecto, (b) es una imagen con fov 80 (c) es una imagen con fov 20 y (d) es una imagen con fov 0.

- Heading (Parámetro opcional): Rumbo de la brújula de la cámara. Los valores aceptados son de 0 a 360 (norte, con el 90 al este y el 180 al sur). Si no se especifica, el rumbo será hacia donde esté la ubicación de *Location*.

  Sobre la esfera representada en la Figura 2.4, la variación de la imagen extraída sería adyacente a la esfera de manera horizontal, como se muestra en la Figura 2.6.

Figura 2.6: Variación del heading de una imagen sobre proyección en una esfera

La Figura 2.7 muestra un ejemplo de 3 imágenes con valor de heading distinto.

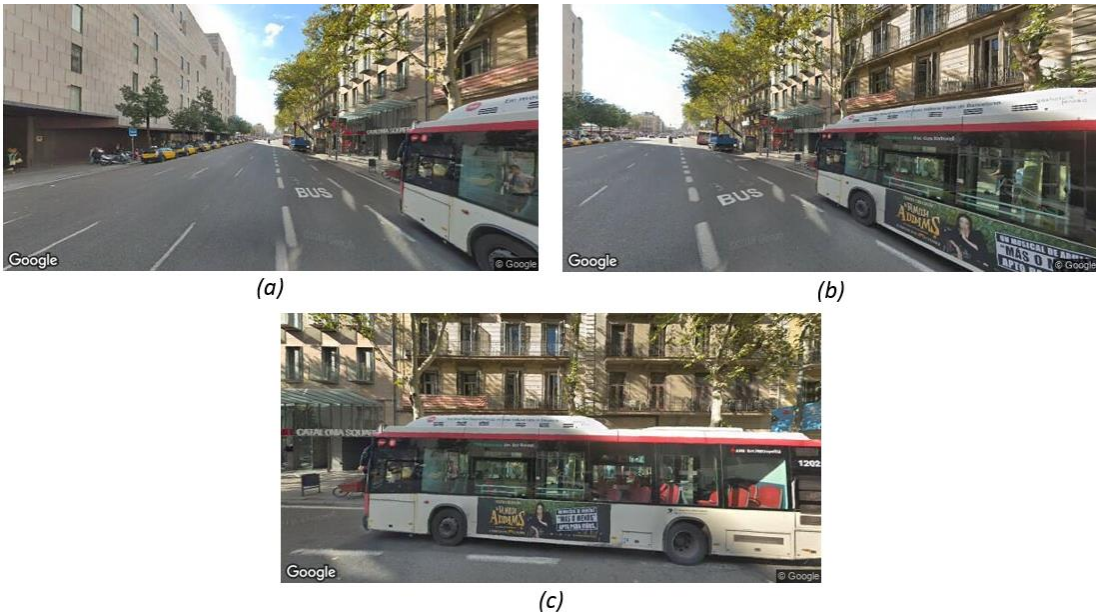

Figura 2.7: Ejemplo de variación del valor del heading en el mismo punto de toma de imágenes donde (a) es una imagen con heading 0, fov 120 y pitch 0, (b) es una imagen con heading 45 y (c) es una imagen con heading 90.

- Pitch (Parámetro opcional): Ángulo hacia arriba o hacia abajo de la cámara en relación con el vehículo de Street View. Esto es a menudo, pero no siempre, el plano horizontal. Los valores positivos inclinan la cámara hacia arriba (hasta 90 grados); los valores negativos inclinan la cámara hacia abajo (hasta 90 grados). El valor por defecto es de 0. El rango de valores aceptados va desde -90 a +90.

  Sobre la esfera representada en la Figura 2.4, la variación de la imagen extraída sería adyacente a la esfera de manera vertical, como se muestra en la Figura 2.8.

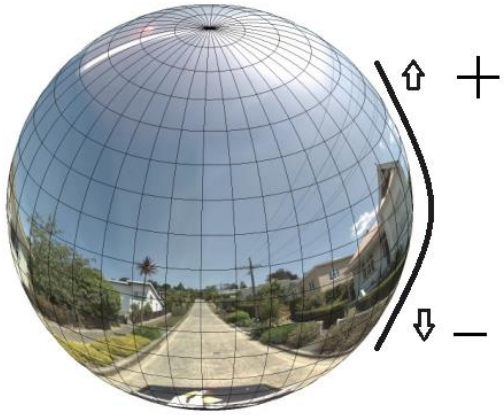

Figura 2.8: Variación del pitch de una imagen sobre proyección en una esfera. Los valores positivos de pitch corresponden a los superiores respecto al ecuador de la esfera, los valores negativos corresponden a los inferiores.

La Figura 2.9 muestra un ejemplo de 3 imágenes con valor de pitch distinto.

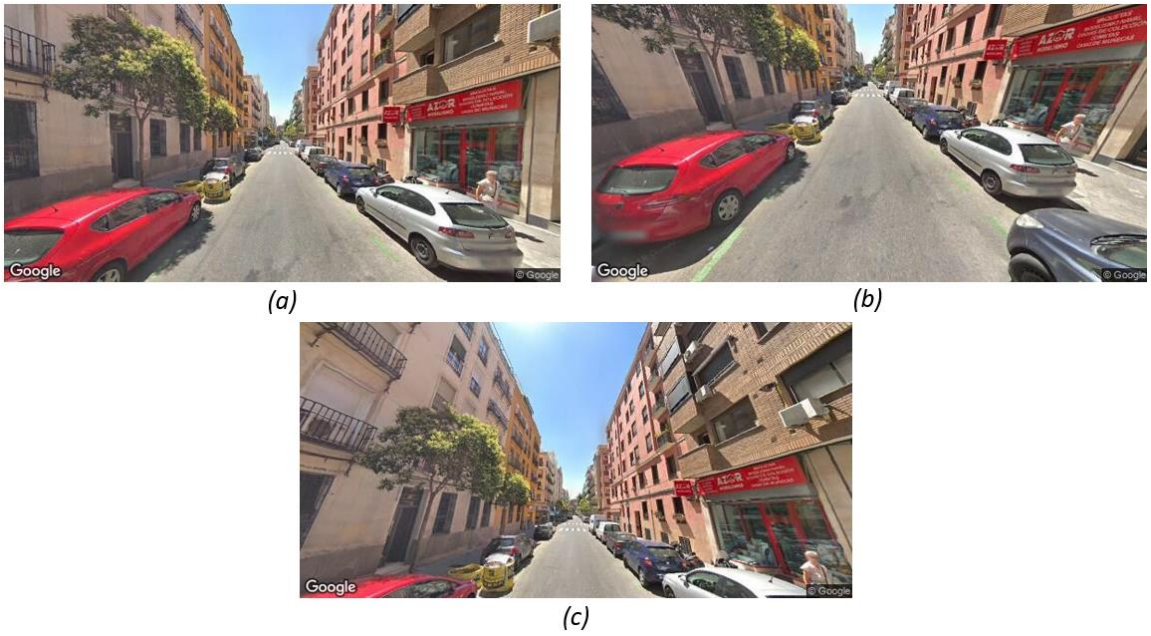

Figura 2.9: Ejemplo de variación del valor del pitch en el mismo punto de toma de imágenes donde (a) es una imagen con pitch 0, fov 120 y heading por defecto, (b) es una imagen con pitch -20 y (c) es una imagen con pitch +20.

- Radius (Parámetro opcional): Radio en metros en los que buscar la imagen según la ubicación. El valor por defecto es 50.

- Source (Parámetro opcional): Limita las búsquedas de las imágenes a imágenes en exteriores.

- DIRECTIONS API [7]

  Biblioteca de funciones, procedimientos y subrutinas que permiten calcular la ruta más eficiente calculando las direcciones entre dos ubicaciones. La ruta es resultado de una solicitud HTTP y una serie de parámetros. Estos parámetros pueden ser obligatorios u opcionales. A continuación, se describen algunos de ellos:

  - Origin (parámetro obligatorio): La dirección, el valor de latitud / longitud o el ID del lugar desde el que se desea calcular las direcciones.

  - Destination (parámetro obligatorio): La dirección, el valor de latitud / longitud o el ID de lugar para el que se desea calcular las direcciones.

  - Key (parámetro obligatorio): Clave API para la utilización de funciones que disponen las APIs de Google. El uso para esta API está limitado a 50 consultas por segundo.

- Mode: Modo de transporte utilizado en la ruta. Dependiendo del modo de transporte, la ruta podrá variar. Se permiten los siguientes:

    - *DRIVING* (Valor por defecto): Calcula las rutas utilizando la red de carreteras.

    - *WALKING*: Calcula las rutas utilizando calles peatonales y aceras.

    - *BYCICLING*: Calcula las rutas utilizando carriles para bicicletas y calles preferentes para bicicletas.

    - *TRANSIT*: Calcula las rutas utilizando trayectos de transporte público.

La Figura 2.10 muestra un ejemplo de 4 imágenes en las que se representa la ruta óptima desde un punto A (Ayuntamiento de Tres Cantos) a un punto B (Biblioteca Municipal de Tres Cantos) variando el modo de transporte. Esta variación hace que la ruta óptima sea diferente para el mismo punto de origen y de destino, dependiendo del modo de transporte.

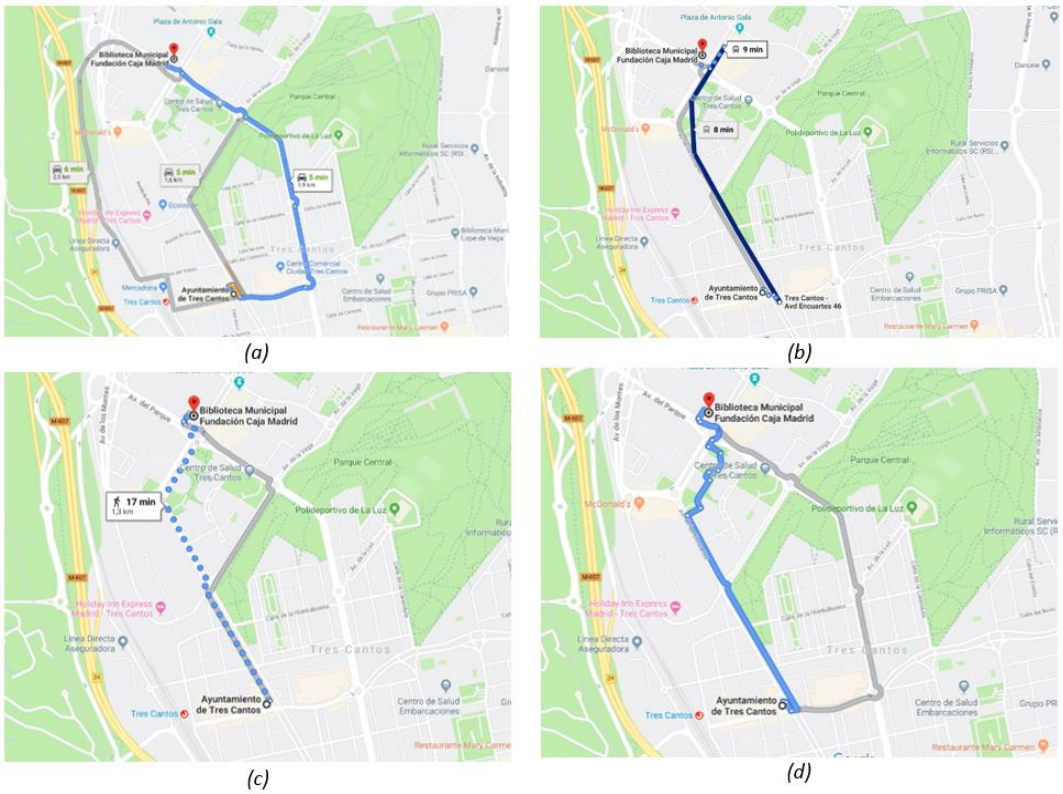

Figura 2.10: Ejemplo de variación del modo de transporte (travel mode) en el mismo punto de toma de imágenes donde (a) es una imagen con ruta óptima (en azul) en coche (travel mode *DRIVING*), (b) en transporte público (travel mode *TRANSIT*), (c) andando (travel mode *WALKING*) y (d) en bicicleta (travel mode *BYCICLING*).

- Waypoints: Matriz de ubicaciones intermedias por las que tiene que pasar o parar la ruta entre los puntos de origen y destino. Se admite solo para los modos de transporte *DRIVING*, *WALKING* y *BYCICLING*.

- Alternatives: Si se establece en true, especifica que se puede proporcionar más de una alternativa de ruta en la respuesta. Se admite solo cuando no hay Waypoints.

- Avoid: Indica que las rutas calculadas deben evitar las siguientes características indicadas:

    - Tolls: La ruta calculada debe evitar carreteras de peaje / puentes.

    - Highways: La ruta calculada debe evitar autopistas.

    - Ferries: La ruta calculada debe evitar los transbordadores.

    - Indoor: La ruta calculada debe evitar los pasos interiores para caminar.

2.1.2 Herramientas existentes analizadas

Para poder cumplir el objetivo de crear una base de datos mediante imágenes de Google Street View a lo largo de rutas específicas, se han evaluado una serie de herramientas existentes, tanto de código libre como cerrado. Estas se desarrollan en distintos lenguajes de programación y presentan diferentes características y limitaciones. Las herramientas evaluadas son las siguientes:

- Pano Fetch. Es una extensión de Google Chrome la cual se puede instalar desde Chrome Web Store en [8]. Descarga una imagen panorámica de la ubicación que se indique en Google Maps. Solo permite la descarga de la imagen indicada, no de varias a lo largo de una ruta definida. Además, el código no es público. Razón por la cual no se puede acceder a él para modificarlo con el objetivo de poder descargar varias imágenes a lo largo de la ruta.

- Street View Download 360. Es una aplicación descargable desde [9]. La imagen que se descarga es una imagen panorámica a través del ID del panorama (diferente para cada imagen de Google Street View). Este ID se puede obtener desde una página web [10] del mismo creador que la aplicación. En esta, indicando una ubicación concreta, la página devuelve directamente la ID de su panorama. Esta página funciona con ayuda de la API key del autor, por lo que no es utilizable, dada su limitación, de forma continuada. Esta aplicación no permite descargar imágenes a lo largo de la ruta, solo imágenes individuales por sus ID. Los códigos de estas dos herramientas descritas tampoco son públicos.

- PanoNom. Es un programa en JavaScript y HTML. Introduciendo la URL, la ID del lugar o la ubicación en longitud y latitud de un lugar específico en Google Street View, representa su respectiva imagen panorámica. El código es público en la plataforma GitHub [11]. Si bien si devuelve las imágenes deseadas, solo devuelve una imagen según la entrada introducida y no imágenes a lo largo de una ruta.

- Google-streetview 1.2.9. Herramienta de línea de comandos que ofrece la propia biblioteca de la API de Google Street View, en el lenguaje de programación de Python [12]. Descarga una imagen indicando como parámetros de una línea de

comandos la clave API, la ubicación en coordenadas geográficas, y otros parámetros como el heading, el fov y el pitch. Solo descarga una imagen a la vez, no imágenes a lo largo de una ruta.

- Google-streetview-images. Es una página web [13] creada por medio de código en lenguaje JavaScript. Introduciendo un origen y un destino busca la mejor ruta entre ellos y recoge las imágenes a lo largo de esta según unos parámetros. Estos parámetros son el número de fotogramas por segundo, el modo de viaje y el tamaño de la imagen a exportar (la imagen no cambia como tal). El código es público en la plataforma GitHub [14]. Se estudió este código para poder utilizarlo y modificarlo con el fin de poder variar los parámetros que nos interesan como el heading, el pitch y el fov. Las imágenes se anexan al DOM (Document Object Model) y no es posible acceder a ellas para descargarlas.

La herramienta que se utilizó finalmente se desarrolla en la Sección 3.1.

2.2 Detección de objetos en imágenes

La detección de objetos es una tecnología que, dada una imagen, detecta diferentes clases de objetos y muestran en la propia imagen su posición. Existen una serie de arquitecturas de modelos de detección de objetos para entrenar, las cuales se introducen en la Sección 2.2.2. Estos algoritmos se entrenan con grandes bases de datos de imágenes con distintas clases de objetos. Estas bases de datos están desarrolladas en el punto 2.2.1.

2.2.1 Algoritmos para la detección de objetos en imágenes

Los algoritmos de detección de objetos, gracias a un previo entrenamiento con un conjunto numeroso de imágenes en las que aparecen representados diferentes clases de objetos, localizan estos objetos en una imagen [15]. Una vez localizados los objetos, el algoritmo devuelve el cuadro delimitador de estos objetos (Bounding Box).

La detección por sí de los objetos no es tarea fácil, ya que, en una imagen, los objetos a buscar pueden estar en cualquier punto de la misma, y tener diferente posición y tamaño. Incluso el objeto puede no aparecer entero (bien porque lo limita la imagen o porque otro objeto lo oculta total o parcialmente). Así pues, las herramientas actuales más utilizadas y que mejores resultados ofrecen están basadas en arquitecturas de redes neuronales convolucionales profundas. Teniendo esto en cuenta, se han desarrollado varios algoritmos con características diferentes para la detección de objetos. De esta forma, podemos encontrarnos algoritmos basados en R-CNN, desarrollado a continuación, los cuales han ido mejorando respecto a la eficiencia computacional, y otros algoritmos con estructuras distintas como son YOLO, SSD, etc.

- R-CNN [16]: Utiliza la búsqueda selectiva (algoritmo fijo) para extraer 2000 regiones (regiones propuestas). Estas regiones en cuadros se incorporan a una red neuronal convolucional. La salida de esta red neuronal es un vector de características de dimensión 4096. Este vector de características produce múltiples probabilidades de pertenencia a cada clase de objecto. Cada clase de objeto tiene un

clasificador SVM entrenado para inducir una probabilidad de detectar ese objeto para un vector de características. Además, también se predicen cuatro valores de desplazamiento para aumentar la precisión del cuadro delimitador. Este proceso se muestra en la Figura 2.11.

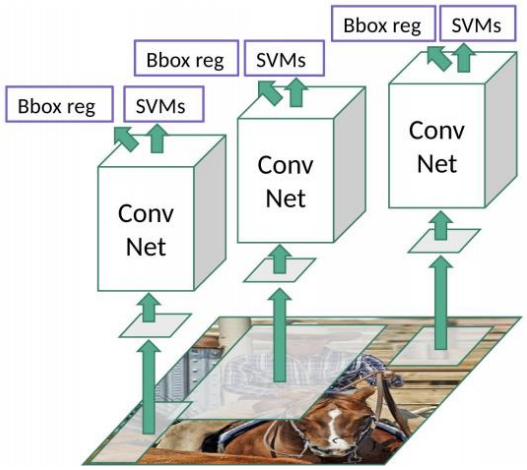

Figura 2.11: Proceso del algoritmo de detección de objetos R-CNN. Fuente: [15]

- Fast R-CNN [17]: El objetivo de este algoritmo es reducir el tiempo de proceso del algoritmo R-CNN. Para ello se realiza la extracción de características una sola vez sobre la imagen entera, y no sobre 2000 regiones propuestas, ejecutando una sola CNN. Con esto obtenemos un mapa de características convolucional. Desde este, se identifican las regiones propuestas y se obtienen vectores de características RoI mediante una capa de agrupaciones de regiones de interés. Por último, en vez de utilizar SVM, se utiliza Softmax para predecir las clases de objectos y los valores de desplazamiento para aumentar la precisión del cuadro delimitador. Este proceso se muestra en la Figura 2.12.

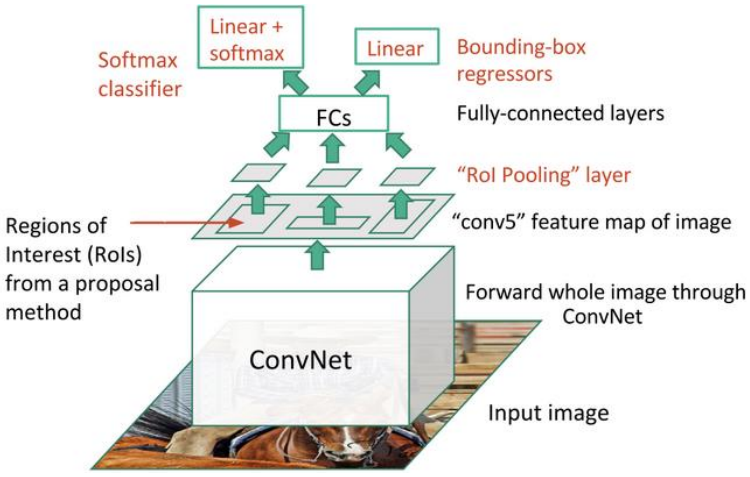

Figura 2.12: Proceso del algoritmo de detección de objetos Fast R-CNN. Fuente: [15]

- Faster R-CNN [18]: Se utiliza RPN (Region Proposal Network) en vez de búsqueda selectiva para generar las regiones propuestas, ya que es mucho más rápido. La

imagen entera es la entrada a la red convolucional y esta, proporciona el mapa de características convolucional (como en Fast R-CNN). Se utiliza RPN y las regiones propuestas se agrupan mediante la capa de agrupación RoI. Esta, clasifica la imagen dentro de la región propuesta, y predice los valores de desplazamiento para los cuadros delimitadores. Este proceso se muestra en la Figura 2.13.

Este algoritmo es tan rápido que puede utilizarse para detecciones de objetos en tiempo real.

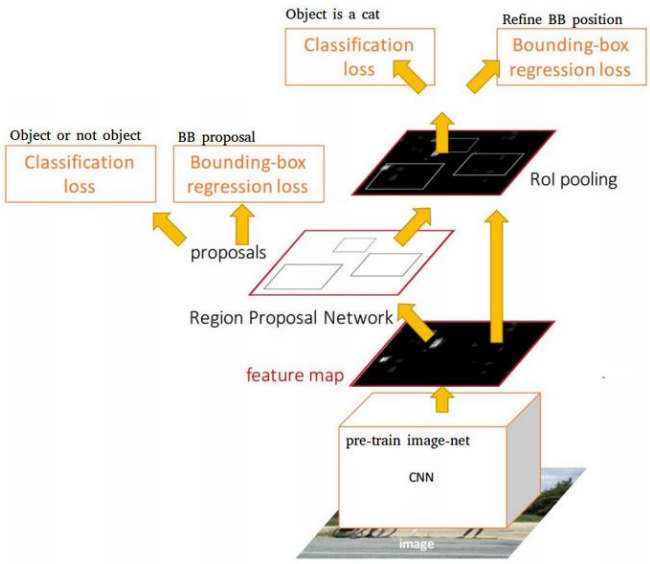

Figura 2.13: Proceso del algoritmo de detección de objetos Faster R-CNN. Fuente: [15]

- YOLO [19]: No es un algoritmo basado en regiones, sino que utiliza la red convolucional directamente para predecir los cuadros delimitadores y sus probabilidades de clases. La imagen de entrada se divide por celdas, en forma de cuadrícula (SxS). Cada celda predice los cuadros delimitadores. Para cada cuadro se calcula una probabilidad de clase y los valores de desplazamiento para aumentar la precisión del cuadro delimitador. Si la probabilidad de clase supera un umbral, el objeto de la clase se detecta como dentro de este cuadro. Este proceso se muestra en la Figura 2.14. Este algoritmo es mucho más rápido que los anteriores.

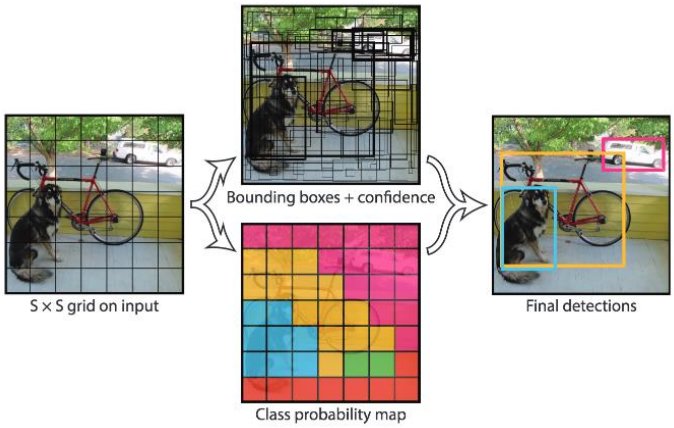

Figura 2.14: Proceso del algoritmo de detección de objetos YOLO. Fuente: [20]

Podemos encontrar diferentes versiones del algoritmo de detección de objetos YOLO, como son:

- YOLOv2 [21]: Es la segunda versión del algoritmo YOLO. Mejora la precisión de este algoritmo y su proceso es más rápido.

- YOLO9000 [21]: Versión del algoritmo YOLO que detecta objetos de más de 9000 clases distintas. Se entrena mediante la base de datos de imágenes de COCO y los clasifica mediante la base de datos de imágenes de ImageNet.

- YOLOv3 [22]: Versión del algoritmo YOLO que utiliza una clasificación de los objetos de varias etiquetas. Es decir, un objeto detectado no lo devuelve como exclusivo de una sola clase.

2.2.2 Bases de datos para entrenar detectores de objetos en imágenes.

Un modelo de detección de objetos sobre imágenes tiene que estar entrenado, antes de ser utilizado, con una serie de imágenes con las clases de objetos que posteriormente va a detectar el algoritmo [23]. Estos modelos una vez pre-entrenados se pueden reentrenar para mejorar las detecciones o añadir más clases de objetos a detectar. Para entrenar un modelo de detección de objetos existen una serie de bases de datos especificas, organizadas por clases. A continuación, se expondrán las bases de datos más importantes y más utilizadas para el entrenamiento de detectores de objetos en imágenes:

- PASCAL Visual Object Classification (VOC) [24]: Base de datos para el reconocimiento de clases de objetos, clasificación y segmentación de objetos. Desde 2005, y cada año hasta 2012, se presentó un desafío nuevo (en total 8 desafíos, ya concluidos) para evaluarlo. La base de datos cuenta con 20 clases de objetos y un total de 10000 imágenes, junto con los cuadros delimitadores de cada objeto en esas imágenes, para el entrenamiento.

- ImageNet [25]: Base de datos para el reconocimiento de clases de objetos, clasificación y segmentación de objetos, en funcionamiento desde 2014. Está organizada de manera jerárquica de acuerdo a WordNet [26], esto es, por el nombre de las clases de objetos. Cuenta con 200 clases de objetos diferentes y un total de 500000 imágenes con los cuadros delimitadores de cada objeto, solo para su entrenamiento.

- COCO [27]: Base de datos para el reconocimiento de clases de objetos, clasificación y segmentación de objetos, desarrollada por Microsoft. Cuenta con 80 clases de objetos diferentes, más de 120000 imágenes para el entrenamiento y 40000 imágenes para pruebas, aunque esta cantidad de imágenes aumenta cada año.

3 Diseño y desarrollo

Podemos dividir este Trabajo de Fin de Grado en dos etapas. Cada etapa tiene una función por separado, aunque el resultado de una es indispensable para la otra. En la **¡Error! No se encuentra el origen de la referencia.** se puede ver un esquema del sistema que forman las dos etapas. Estas dos etapas son:

- Primera etapa: Desarrollo de una base de datos a partir de imágenes a lo largo de rutas de Google Street View
- Segunda etapa: Detección de objetos a partir de las imágenes de la base de datos creada.

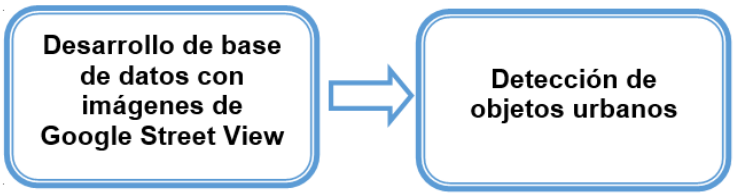

Figura 3.1: Sistema del procedimiento del Trabajo de Fin de Grado.

El objetivo de este capítulo es desarrollar las tecnologías utilizadas para realizar cada etapa. En la Sección 3.1 se desarrollan las tecnologías de la primera etapa. En la Sección 3.2 se desarrollan las tecnologías para la segunda etapa.

3.1 Herramienta StreetView-Extractor

StreetView-Extractor es una herramienta basada en un código abierto en la plataforma GitHub [28] desarrollado en el lenguaje de programación Java. Para poder construir el proyecto se utiliza la herramienta de software Apache Maven.

Este programa descarga las imágenes a lo largo de una ruta especificada, un video creado a partir de ellas y un archivo json. Este archivo json contiene el conjunto de solicitudes HTTP para la extracción de las imágenes en la ruta. Para poder realizar esto, se ejecuta el programa junto con una serie de parámetros, desarrollados en la siguiente Sección 3.1.1.

Originalmente el código solo permitía introducir como parámetros la clave API (parámetro obligatorio), la dirección de origen de la ruta y la de destino (parámetros obligatorios), el número de imágenes por metro o por segundo, el tamaño de la imagen, y la opción de activar la descarga de imágenes, video y el archivo json.

Para optimizar el resultado, se actualizaron las dependencias del archivo pom.xml con el número de versión correcto para un buen funcionamiento del programa. Además, se modificó el código para poder variar otros parámetros que originalmente eran fijos, como son el modo de la ruta, el ángulo de la cámara tanto horizontal (heading) como

verticalmente (pitch), y el campo de visión (fov). El código definitivo descarga imágenes eficientemente a lo largo de una ruta con una variación muy flexible de parámetros.
En el anexo A se incluye un manual de este programa.

Para obtener diferentes versiones de una misma ruta variando cada parámetro, se creó un programa en lenguaje Python. Este programa hace un barrido de los parámetros y utiliza con estos la herramienta de extracción de imágenes (StreetView-Extractor). Esto se repite para distintas rutas. Así obtenemos finalmente una base de datos con varias rutas y versiones de las mismas.

3.1.1 Parámetros

Para poder extraer las imágenes de cada ruta se necesita introducir una serie de parámetros a la llamada de la línea de ejecución. Se utilizan varios de los parámetros de las APIs de la Sección 2.1.1 además de otros añadidos:

- ORIGEN: Parámetro de Directions API, desarrollado en la Sección 2.1.1
- DESTINO: Parámetro de Directions API, desarrollado en la Sección 2.1.1
- API KEY: Parámetro de Directions API y Street View API, desarrollado en la Sección 2.1.1
- FOV: Parámetro de Street View Static API, desarrollado en la Sección 2.1.1. El valor por defecto es de 90.
- PITCH: Parámetro de Street View Static API, desarrollado en la Sección 2.1.1.
- TRAVEL MODE: Parámetro de Directions API, desarrollado en la Sección 2.1.1.
- HEADING: Parámetro de Street View Static API, desarrollado en la Sección 2.1.1. El valor por defecto es 361.
- HEAD: Puede establecerse en *true* o en *false*. Si se establece en *true,* el rumbo de la brújula de la cámara que tomó la imagen (HEADING) será el que se dirige hacia el siguiente punto de toma de imágenes en la ruta. En función del valor de Heading seleccionado se podrán obtener 2 resultados:
    - Si el HEADING no se indica (valor por defecto (361)) el rumbo o dirección de la imagen será el indicado en primer término.
    - Si el HEADING se indica, el rumbo resultante será la suma del rumbo obtenido en primer término más el valor de HEADING indicado.
- Nombre salida archivo json: Nombre del archivo json. El archivo solo es creado si no existe alguno con el mismo nombre.
- Salida de imágenes: Puede establecerse en *true* o en *false*. Si se establece en *true* se devolverán una serie de imágenes a lo largo de la ruta.

- Salida de video: Puede establecerse en *true* o en *false*. Si se establece en *true* se devolverá un video recopilando todas las imágenes de la ruta.

- Tamaño de la imagen: Parámetro de Street View Static API, desarrollado en la Sección 2.1.1. El valor por defecto es de 300x600.

- FPX: Número de imágenes por X. Siendo X valor de referencia en distancia (metros) o en tiempo (segundos). Si la opción TIME-RECODE se establece en *true*, FPX es el número de imágenes por segundo. Si se establece en *false* FPX es el número de imágenes por metro. El valor por defecto es de 0.1 imágenes por X.

- TIME-RECODE: Si se establece en *true*, el parámetro FPX será número de imágenes por segundo. Por tanto, las imágenes serán de posiciones más separadas cuanto más rápido vaya el vehículo que las tomó.

En el anexo A se indica cómo ejecutar el programa con estos parámetros.

3.2 Detección de objetos sobre las imágenes urbanas

Una vez creada la base de datos con imágenes urbanas a lo largo de rutas, se procedió a la prueba de un detector de objectos sobre ellas.

El Modelo elegido para la detección de objetos sobre nuestra base de datos es el algoritmo YOLOv3, descrito en la Sección 2.2.1 [29]. Se eligió este modelo de detector ya que destaca por su velocidad, debido al bajo coste computacional, y su alta eficiencia. Es el detector ideal para nuestro trabajo puesto que se prueba sobre un conjunto grande de imágenes, por lo que necesitamos que la detección sea rápida.

El algoritmo YOLOv3, utilizado sobre nuestra base de datos, fue entrenado con la base de datos de entrenamiento de COCO, ya que esta es lo suficientemente amplia para que la detección tenga buena capacidad de generalización. Cuenta con 80 clases de objetos de los cuales encontramos los que nos interesan para detectar en las imágenes de nuestra base de datos.

El proceso de detección de objetos sobre las imágenes urbanas de nuestra base de datos que se realizó es el siguiente:

1. Listado de las imágenes de nuestra base de datos. La lista incluirá las imágenes de cada ruta que queramos que se procesen con el detector YOLOv3, para la detección de los objetos solicitados.

2. Ejecución el detector YOLOv3 sobre las imágenes de nuestra base de datos. Se entrega al detector como entrada la lista de imágenes de cada ruta a detectar creada en el paso 1. Las detecciones de cada imagen en la ruta se agrupan en resultados según la clase de objeto detectada. Así, para cada ruta de imágenes, obtenemos los resultados agrupados según las clases. En los resultados se incluyen el nombre de la imagen en la ruta, donde se detectó esa clase de objeto, el cuadro delimitador de la detección y la precisión de clase de detección del objeto.

3. Representación de los resultados de las detecciones en las imágenes. Mediante Matlab se representan e insertan sobre las imágenes procesadas los respectivos resultados de la detección de objetos para cada una. Se inserta y representa solo aquellos resultados con una precisión de clase mayor a un umbral, distinto para cada clase.

3.2.1 Evaluación del detector

Como parte del proceso de detección se realiza una evaluación de esta. Para ello se crea un Ground Truth para un conjunto de imágenes mediante la aplicación que proporciona para ello Matlab, llamada Image Labeler. Esta aplicación permite posicionar a mano cada clase de objeto en las imágenes. El Ground Truth está formado por esta información sobre la posición óptima de cada clase en las imágenes. Se creó un programa para evaluar los resultados de las detecciones del conjunto de imágenes escogido, en comparación con el Ground Truth de estas imágenes. Para ello, se utilizó la función de Matlab *evaluateDetectionPrecision.m* [30]. Esta función devuelve puntos de datos para trazar la curva de precisión-recall y la precisión media, utilizando como argumentos de entrada el Ground Truth y los resultados de las detecciones.
Posteriormente, el programa calcula la curva de precisión-recall.

La curva de Precision-Recall para detección de objetos es una representación de la evaluación del rendimiento de un detector de objetos. [31] Relaciona la precisión con la sensibilidad del modelo de detección a la hora de clasificar las detecciones como de una clase de objeto o de otra. Para ello, nos podemos encontrar las siguientes clasificaciones:

- TP (True Positive): Se detecta y clasifica como objeto de la clase A, a un objeto de la clase A en la imagen. El detector ha acertado.

- FP (False Positive): Se detecta y clasifica como objeto de la clase A, a un objeto de la clase B o a ningún objeto en la imagen. El detector ha errado.

- TN (True Negative): No se detecta ningún objeto de la clase A, y no hay ningún objeto de la clase A en la imagen. El detector ha acertado.

- FN (False Negative): No se detecta ningún objeto de la clase A, pero si hay objetos de la clase A en la imagen. El detector ha errado.

Mediante estos conceptos, se pueden hallar la precisión (precision) y la sensibilidad (recall) de la siguiente manera:

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

Un modelo ideal tendría tanto una alta precisión como una alta sensibilidad.

Para una mejor evaluación también se calcula la precisión media (average precision). Esta es el área bajo la curva, es decir, su integral. La precisión media de un modelo ideal es cercana a 1.

4 Integración, pruebas y resultados

Una vez explicadas las tecnologías y procedimientos para la creación de una base de datos mediante imágenes de Google Street View a lo largo de rutas y la detección de objetos sobre esta, se procede a desarrollar los resultados de ambas fases. En la Sección 4.1 se desarrollan los resultados de la creación de nuestra base de datos y en la Sección 4.2 se desarrollan los resultados de las detecciones sobre esta.

4.1 Resultados de la creación de la base de datos

Para la creación de nuestra base de datos se utilizó el programa que automatiza la extracción de imágenes mediante la herramienta StreetView-Extractor, desarrollada en la Sección 3.1. Este programa ejecuta la herramienta realizando un barrido de los parámetros. Se utilizaron unos determinados parámetros para la extracción de las imágenes de nuestra base de datos, de todos los posibles (desarrollados en la Sección 3.1.1). Estos parámetros utilizados para la creación de nuestra base de datos se muestran en la siguiente sección 4.1.1.

En cuanto a las diferentes rutas utilizadas para la extracción de imágenes a lo largo de estas, se muestran en la Sección 4.1.2, resultando de estas, las imágenes de nuestra base de datos.

4.1.1 Parámetros utilizados

Para crear nuestra base de datos hemos utilizado unos parámetros de origen y destino para cada ruta, nuestra propia api key, la salida de imágenes y head activados, los modos de transporte *DRIVING* y *WALKING* y una variación, expuesta en la Tabla 4.1 para un heading de 361 y en la Tabla 4.2 para un heading de +90, de pitch y de fov. Los demás parámetros toman valor por defecto.

HEADING	361												
FOV	20	40	60	80	100	120							
PITCH	0	0	0	0	0	-40	-30	-20	-10	0	10	20	30

Tabla 4.1: Combinación de los parámetros para un HEADING de 361, para cada TRAVEL MODE (*DRIVING* y *WALKING*)

HEADING	+90									
FOV	60	80	100	120						
PITCH	0	0	0	-40	-30	-20	-10	0	10	20	30

Tabla 4.2: Combinación de los parámetros para un HEADING de +90, para cada TRAVEL MODE (*DRIVING* y *WALKING*)

En total obtenemos un total de 48 combinaciones distintas de parámetros para una misma ruta (mismo origen y destino de la ruta).

4.1.2 Rutas recogidas

A la hora de crear nuestra base de datos para poder aplicar después un detector de objetos urbanos sobre ella, se buscaron diferentes localidades y espacios urbanos, cada uno con propiedades y objetos diferentes. Con este objetivo se han seleccionado rutas variadas siendo estas las siguientes:

- Ruta 1: Obtenida de la ciudad Tres Cantos, Madrid. Está formada por 173 imágenes en modo *DRIVING* y 86 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (a).

- Ruta 2: Obtenida de la ciudad Madrid, España (zona SOL - Gran Vía). Está formada por 230 imágenes en modo *DRIVING* y 49 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (b).

- Ruta 3: Obtenida de la ciudad Madrid (barrio Chamberí). Está formada por 30 imágenes tanto en modo *DRIVING* como en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (c).

- Ruta 4: Obtenida de la ciudad Alcorcón, Madrid. Está formada por 65 imágenes en modo *DRIVING* y 26 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (d).

- Ruta 5: Obtenida de la ciudad Copenhague (Dinamarca). Está formada por 103 imágenes en modo *DRIVING* y 39 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (e).

- Ruta 6: Obtenida de la ciudad Barcelona (zona Portal del Ángel). Está formada por 265 imágenes en modo *DRIVING* y 54 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (f).

- Ruta 7: Obtenida de Chinchón, pueblo de Madrid. Está formada por 100 imágenes en modo *DRIVING* y 57 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (g).

- Ruta 8: Obtenida de Daca (Bangladesh). Está formada por 182 imágenes en modo *DRIVING* y 117 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (h).

- Ruta 9: Obtenida de Nueva York (EEUU) zona Times Square. Está formada por 96 imágenes tanto en modo *DRIVING* como en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (i).

- Ruta 10: Obtenida de Nom Pen (Camboya). Está formada por 196 imágenes en modo *DRIVING* y 157 imágenes en modo *WALKING*. Se muestra un ejemplo de esta ruta en la Figura 4.1 (j).

El entorno de las rutas es variado. Incluye entornos urbanos y rurales, calles amplias y estrechas, despejadas y cargadas de objetos, occidentales y orientales. Los objetos que podemos encontrar en ellas son personas, coches, autobuses, camiones, motocicletas, bicicletas, carros, señales de tráfico, semáforos, parquímetros, bancos, bolsos y maletas.

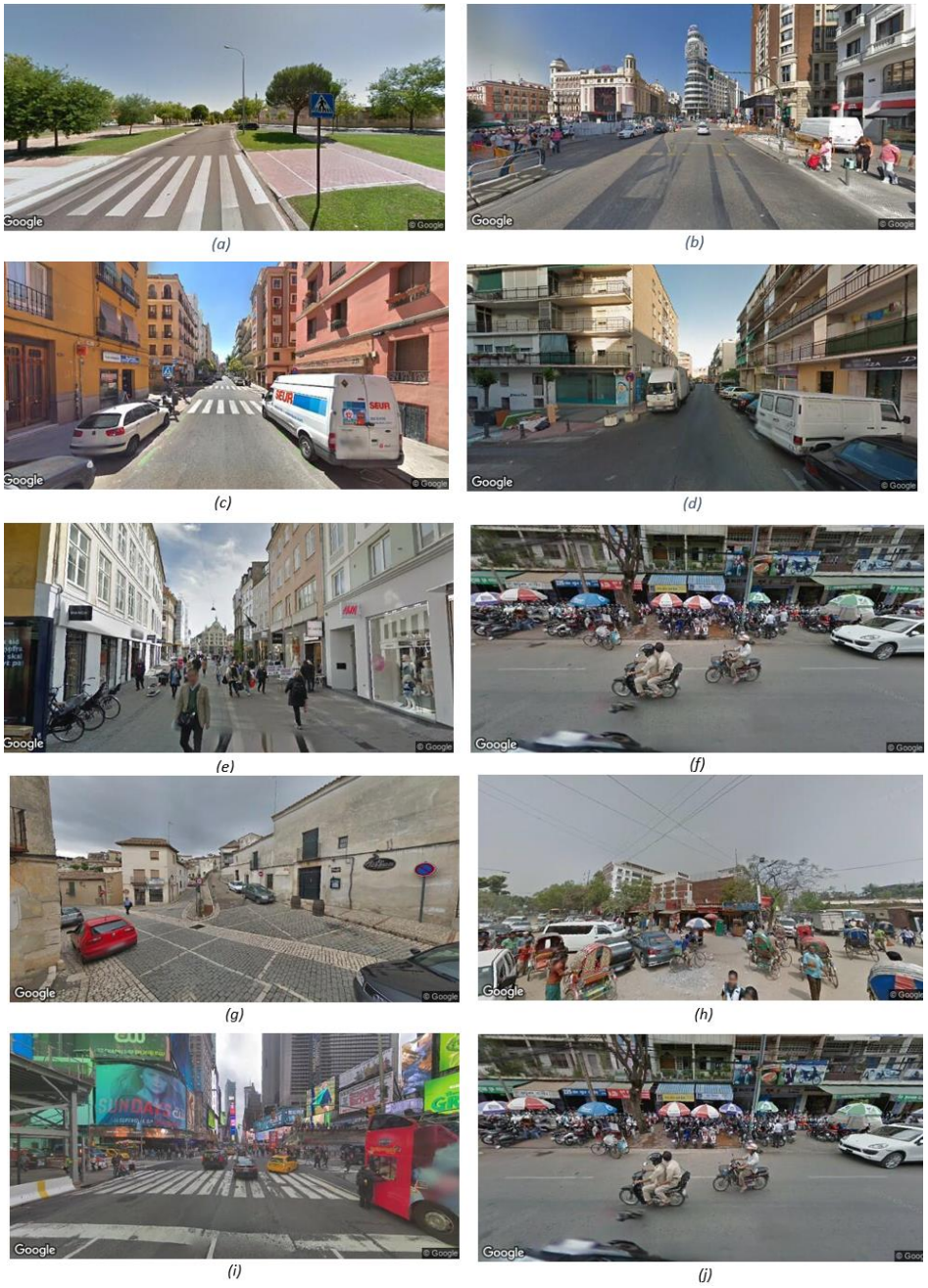

Figura 4.1: Ejemplos de imágenes de rutas donde (a) es extraída de la ruta 1, (b) es extraída de la ruta 2, (c) es extraída de la ruta 3, (d) es extraída de la ruta 4, (e) es extraída de la ruta 5, (f) es extraída de la ruta 6, (g) es extraída de la ruta 7, (h) es extraída de la ruta 8, (i) es extraída de la ruta 9 y (j) es extraída de la ruta 10, cada una con parámetros variables distintos.

Es observable que algunas imágenes están distorsionadas y deformadas, sobre todo por la zona inferior de estas. Esto se debe a que las imágenes son extraídas como subimagenes de una imagen panorámica de la base de datos de Google Street View, como se explicó en la Sección 2.1.1. Las imágenes panorámicas de la base de datos de Google Street View son formadas mediante un proceso de stitching, juntando imágenes no panorámicas que representan todo el espacio en 360 grados. Las imágenes no panorámicas que utilizan para formar la imagen total panorámica, a veces no son tomadas en el mismo instante o alguna de las cámaras no está bien enfocada. Esto produce que al combinarlas se originen cortes y deformaciones en la imagen final. Las imágenes de la **¡Error! No se encuentra el origen de la referencia.** (e) y (j) muestran un ejemplo de este efecto.

Otra observación notable es la deformación geométrica de los objetos que encontramos en las imágenes, según el ángulo vertical (pitch) con el que se extrae la foto. Se da sobre todo en los objetos más cercanos al punto de toma de las imágenes. La forma de los objetos se deforma más cuanta más variación de pitch tenga. En las siguientes figuras se puede observar un ejemplo de este efecto. La imagen (a) de la Figura 4.2 es tomada con un valor de pitch 0. La imagen (b) de la Figura 4.2 es extraída con un valor de pitch de -20. Se observa una deformación en sus objetos, sobre todo en el vehículo más cercano. En la imagen (c) de la Figura 4.2 se puede apreciar notablemente la deformación y la variación de esta respecto a la de la imagen (a). Estas imágenes con deformaciones geométricas distintas en sus objetos se recogen con la intención de una mejor evaluación del detector, en la Sección 4.2.1.

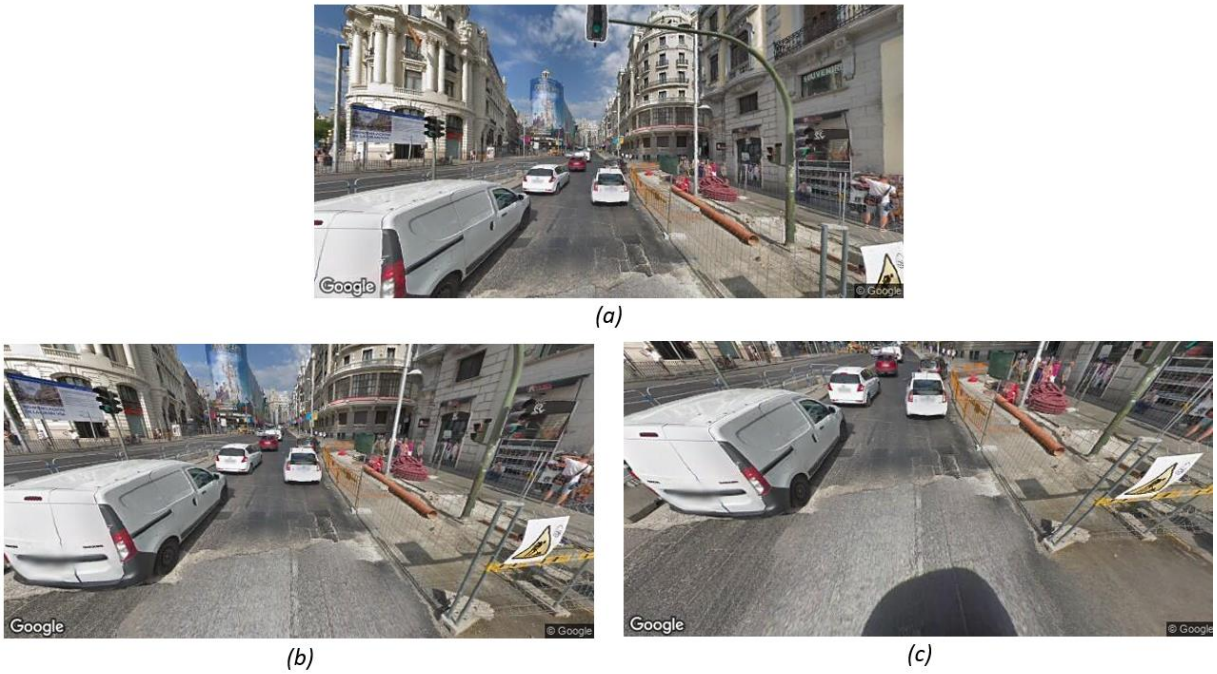

Figura 4.2: Ejemplo de deformación geométrica de los objetos que se da en una imagen debida a la variación del pitch. (a) es una imagen con valor de pitch 0. (b) es una imagen con valor de pitch -20. (c) es una imagen con valor de pitch -40.

4.2 Detección de objetos sobre la base de datos

Una vez creada nuestra base de datos de imágenes urbanas de Google Street View a lo largo de cada ruta, se procedió a probar un detector de objetos sobre ella. El proceso de esta etapa se desarrolla en la Sección 3.2.

Los resultados de la detección YOLOv3 en las imágenes de cada ruta (paso 2 del proceso de detección en la Sección 3.2) se obtuvieron en documentos de texto para cada clase a detectar.

A la hora de insertar estos resultados en las imágenes correspondientes, explicado en el paso 3 del proceso de detección (Sección 3.2), se tuvieron en cuenta solo algunos resultados para algunas clases de objetos, de todas las clases con las que está entrenado nuestro detector mediante COCO (80 clases de objetos). Estos resultados son los correspondientes a las clases de objetos que podemos encontrar en un entorno urbano: personas, coches, autobuses, bicicletas, motocicletas, camiones, bancos, parquímetros, señales de stop, semáforos, bolsos y maletas. Se eligieron solo estas 12 clases de objetos ya que son las más probables de encontrar en entornos urbanos de entre todas las clases que puede detectar el detector YOLOv3 utilizado.

Cuando se trata de insertar las detecciones de estas clases en las imágenes, para obtener los resultados visuales, solo aquellas con una precisión mayor a un umbral se tuvieron en cuenta. Estos umbrales de precisión fueron elegidos subjetivamente estudiando las detecciones resultantes para cada clase de objeto.

Los cuadros delimitadores que señalan las detecciones insertadas en las imágenes son de distintos colores según la clase de objeto que hayan detectado.

La Tabla 4.3 muestra los umbrales de precisión y los colores de los cuadros delimitadores a la hora de representarlo, elegidos para cada clase de objeto.

Clase de objeto	Umbral de precisión	Color en la representación
Personas	0.4	Rojo
Coches	0.3	Azul
Autobuses	0.4	Verde claro
Bicicletas	0.2	Cian
Motocicletas	0.2	Magenta
Camiones	0.3	Amarillo
Bancos	0.2	Negro
Parquímetros	0.2	Naranja
Señales de stop	0.4	Gris
Semáforos	0.4	Verde oscuro
Bolsos	0.2	Blanco
Maletas	0.2	Marrón

Tabla 4.3: Umbrales de precisión y colores de los cuadros delimitadores escogidos para representar cada clase de objeto detectado.

Las imágenes resultan de las imágenes de nuestra base de datos original con sus detecciones correspondientes insertadas en ellas. Estas detecciones se representan mediante los cuadros delimitadores de cada color para cada clase, y su precisión. La Figura 4.3, la Figura 4.4 y la Figura 4.5 son un ejemplo.

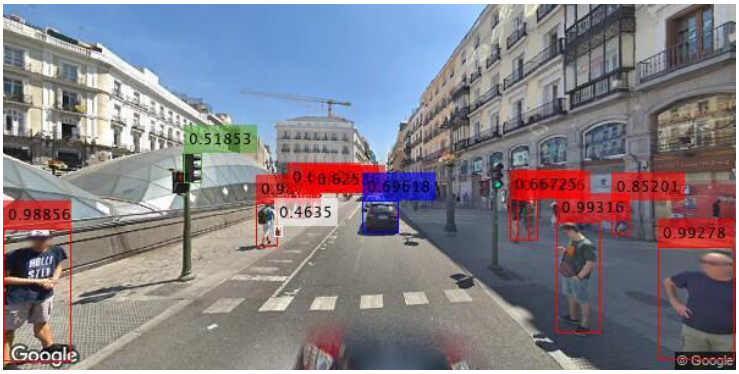

Figura 4.3: Imagen resultante de la detección de objetos. Imagen original extraída de la ruta 2, mode DRIVING, heading por defecto, pitch 0 y fov 120.

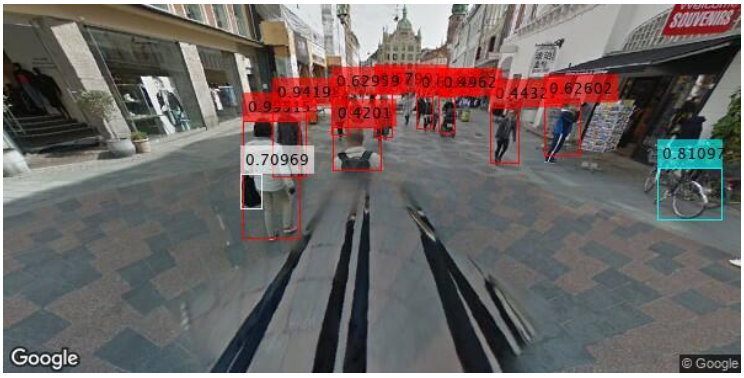

Figura 4.4: Imagen resultante de la detección de objetos. Imagen original extraída de la ruta 5, mode WALKING, heading por defecto, pitch -30 y fov 120.

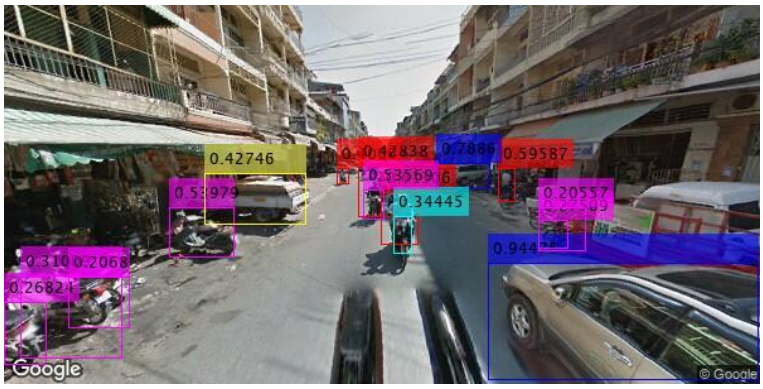

Figura 4.5: Imagen resultante de la detección de objetos. Imagen original extraída de la ruta 10, mode DRIVING, heading por defecto, pitch -10 y fov 120.

Como se puede observar, los objectos que aparecen no enteros en las imágenes, ya sea por encubrimiento de otros objetos, por las distorsiones de la imagen o por los propios límites de esta, pueden ser detectados eficazmente.

También es observable que las distorsiones que se forman a veces en el área inferior de las imágenes no alteran la detección de objetos.

4.2.1 Evaluación de los resultados

Una vez realizadas las detecciones se procede a una primera evaluación de estas. Para ello se realiza el proceso descrito en la Sección 3.2.1. A la hora de realizar la evaluación se seleccionaron imágenes con variedad de objetos, pero sin un número excesivo de estos. Esto es porque las imágenes donde encontramos numerosos objetos, el etiquetado a mano para realizar el Ground Truth puede ser tedioso y no riguroso. Por esta razón, para la realización del Ground Truth (con Image Labeler), se utilizaron imágenes con un número de objetos en ellas no muy alto.

La evaluación se realizó sobre varios grupos de imágenes de distintas rutas. Además, para una mejor evaluación del detector, se probó sobre imágenes con distintas deformaciones geométricas, debido al pitch, como se explicó en la Sección 4.1.2. Como resultado de la evaluación, para cada conjunto de imágenes, se representaron las curvas de Precisión – Recall para cada clase de objeto, para distintos valores de pitch, los cuales son 0, -20, -40. También se calcularon las precisiones medias.

A continuación, se exponen los diferentes grupos de imágenes que se evaluaron y los resultados de las evaluaciones que se realizaron:

- Conjunto de imágenes 1. 14 imágenes de la ruta 2. Las curvas de Precision - Recall de la detección en este conjunto de imágenes, para las clases de objetos más numerosas (personas y coches) y, por ello, con curvas más destacadas, para cada valor de pitch, se representan en la **¡Error! No se encuentra el origen de la referencia.**. Las respectivas precisiones medias se exponen en la Tabla 4.4.

- Conjunto de imágenes 2. 30 imágenes de la ruta 3. Las curvas de Precision - Recall de la detección en este conjunto de imágenes, para las clases de objetos más numerosas (personas, coches, motocicletas y camiones) y, por ello, con curvas más destacadas, para cada valor de pitch, se representan en la **¡Error! No se encuentra el origen de la referencia.**. Las respectivas precisiones medias se exponen en la Tabla 4.5. Según estos resultados, se puede observar que las mejores detecciones se dan para las clases de objetos de coches, ya que son la clase más numerosa en las imágenes.

- Conjunto de imágenes 3. 25 imágenes de la ruta 5. Las curvas de Precision - Recall de la detección en este conjunto de imágenes, para las clases de objetos más numerosas (personas, coches y bicicletas) y, por ello, con curvas más destacadas, para cada valor de pitch, se representan en la **¡Error! No se encuentra el origen de la referencia.**. Las respectivas precisiones medias se exponen en la Tabla 4.6.

Según estos resultados, se puede observar que las mejores detecciones se dan para las clases de objetos de coches, ya que son la clase más numerosa en las imágenes.

- Conjunto de imágenes 3. 15 imágenes de la ruta 9. Las curvas de Precision - Recall de la detección en este conjunto de imágenes, para las clases de objetos más numerosas (personas y coches) y, por ello, con curvas más destacadas, para cada valor de pitch, se representan en la **¡Error! No se encuentra el origen de la referencia.**. Las respectivas precisiones medias se exponen en la Tabla 4.7. Según estos resultados, se puede observar que las mejores detecciones se dan para las clases de objetos de coches, ya que son la clase más numerosa en las imágenes.

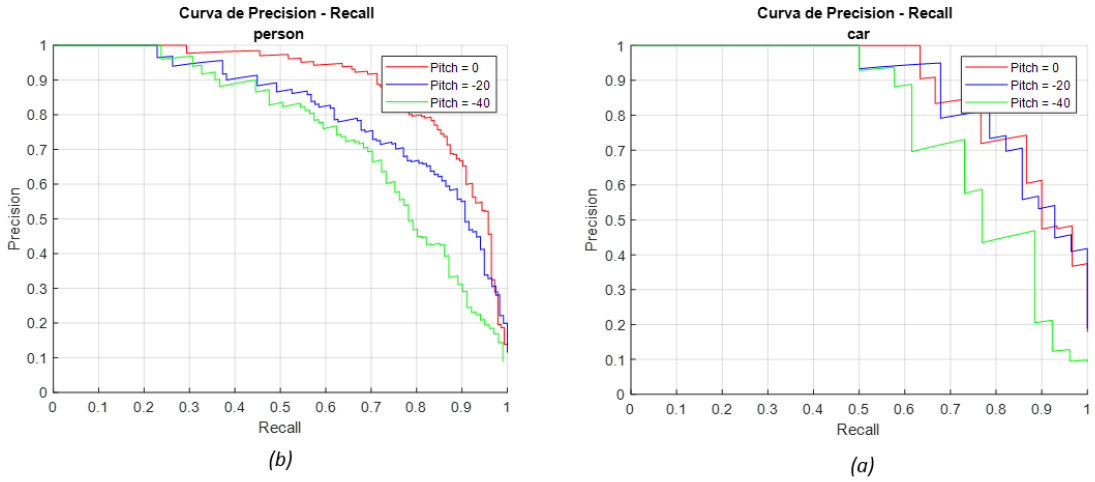

Figura 4.6: Curvas de Precision-Recall del conjunto de imágenes 1 variando el valor del pitch, siendo 0, -20 y -40, para (a) clase de objetos personas y (b) clase de objetos coches.

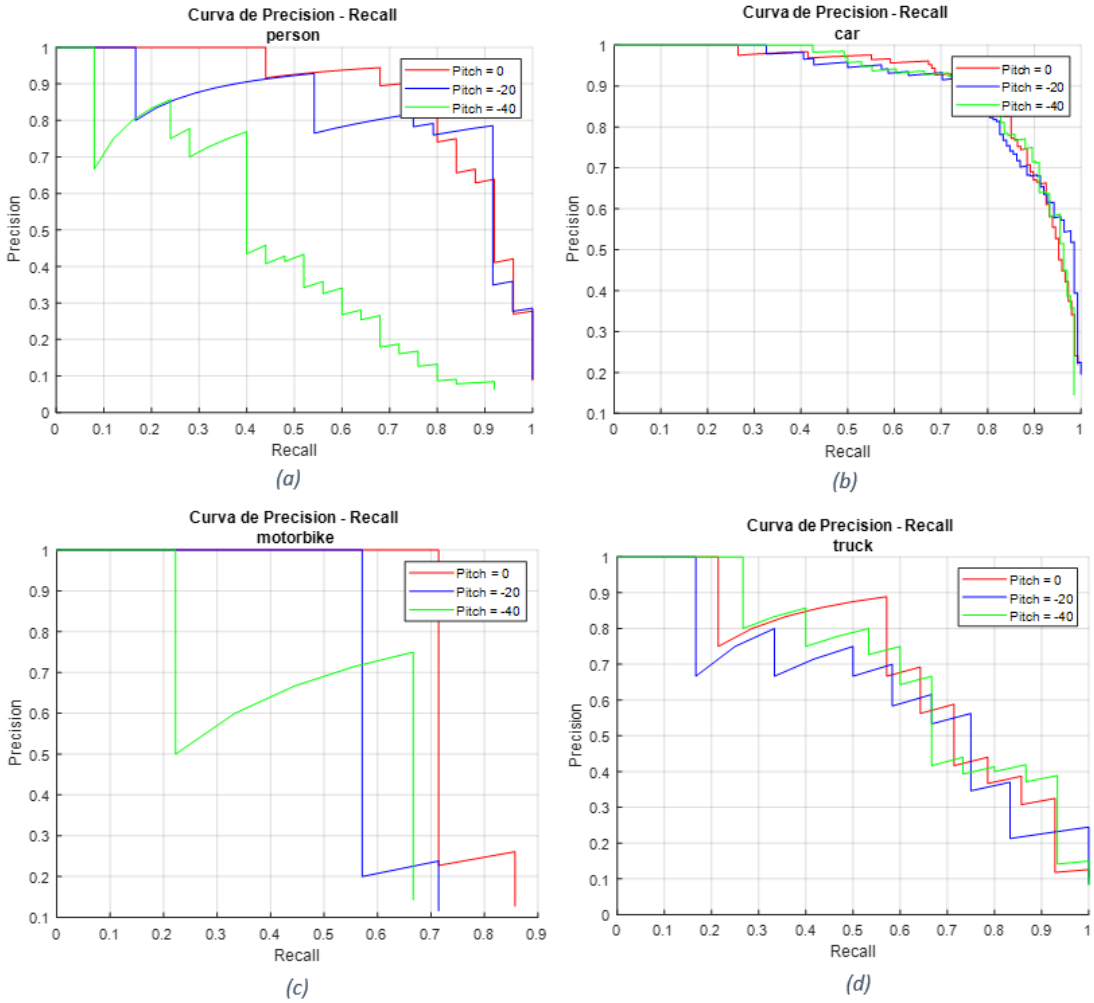

Figura 4.7: Curvas de Precision-Recall del conjunto de imágenes 2 variando el valor del pitch, siendo 0, -20 y -40, para (a) clase de objetos personas, (b) clase de objetos coches, (c) clases de objetos motocicletas y (d) clase de objetos camiones.

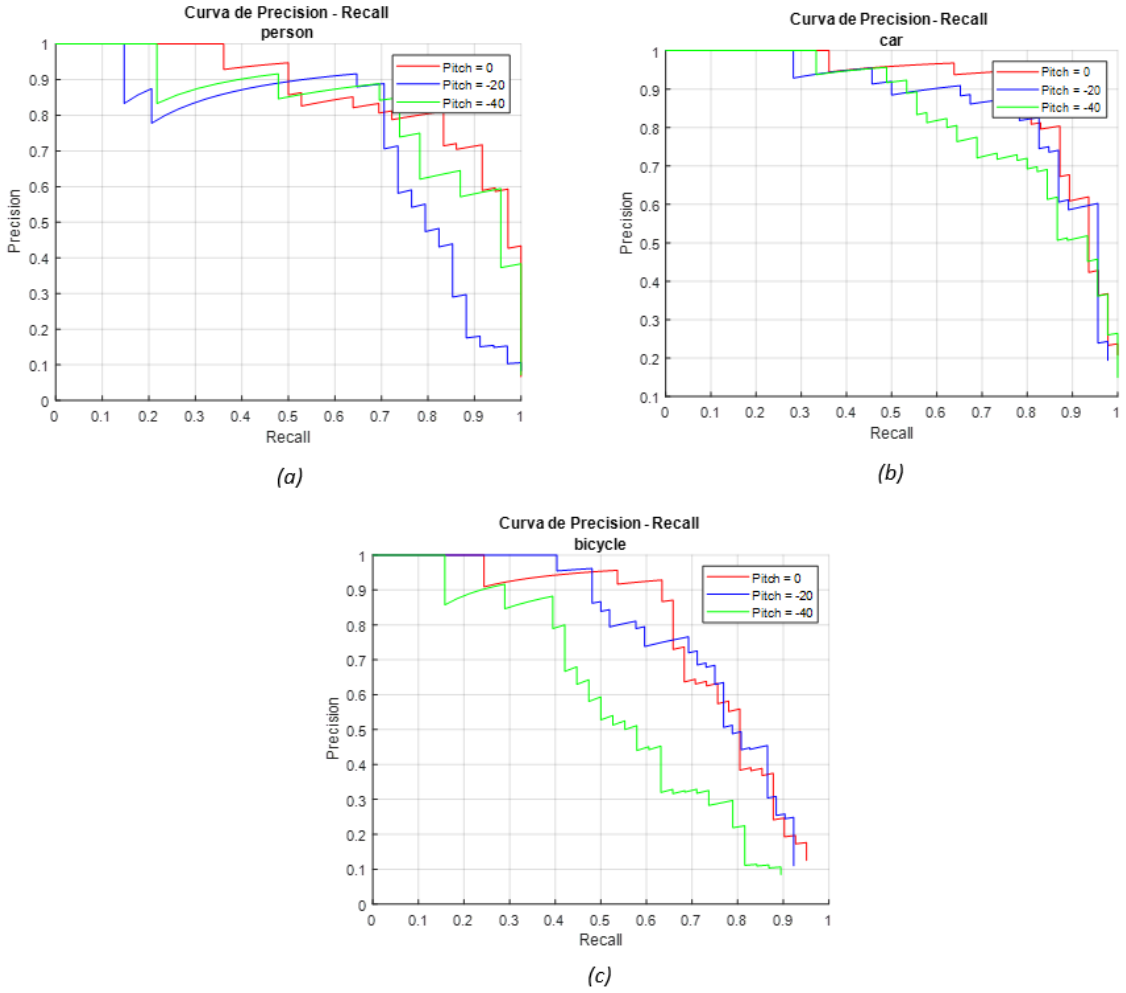

Figura 4.8: Curvas de Precision-Recall del conjunto de imágenes 3 variando el valor del pitch, siendo 0, -20 y -40, para (a) clase de objetos personas, (b) clase de objetos coches y (c) clases de objetos motocicletas.

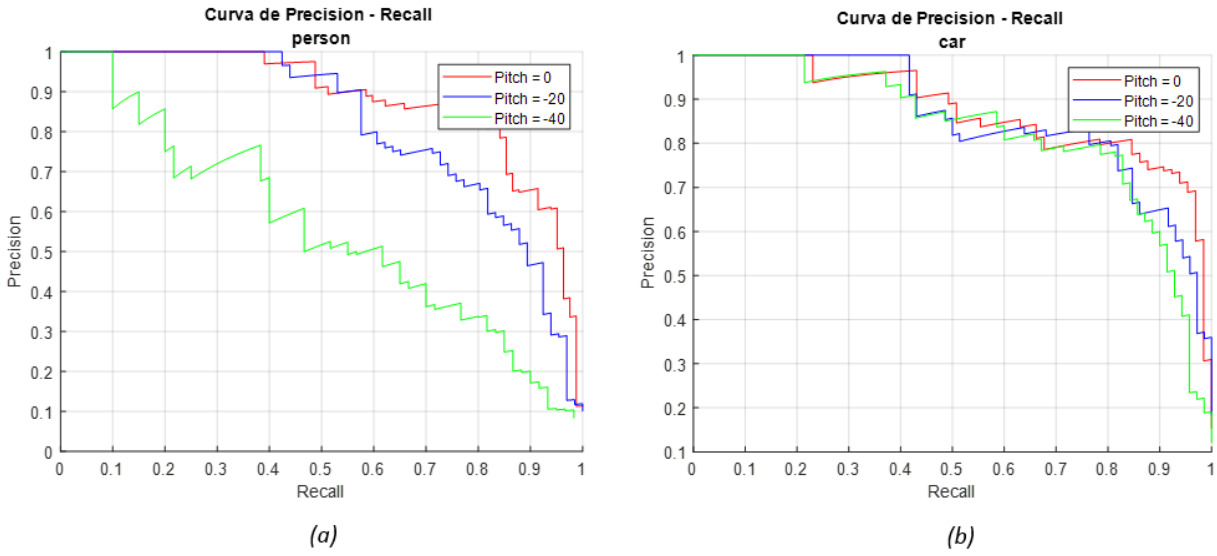

Figura 4.9: Curvas de Precision-Recall del conjunto de imágenes 4 variando el valor del pitch, siendo 0, -20 y -40, para (a) clase de objetos personas y (b) clase de objetos coches.

PITCH / CLASE DE OBJETOS	Personas	Coches
0	0.88952	0.88691
-20	0.81461	0.87772
-40	0.74932	0.78154

Tabla 4.4: Precisiones medias de la evaluación del grupo de imágenes 1, en función del pitch con el que está tomado la imagen de la que se evalúa su detección y la clase de objeto de la detección evaluada.

PITCH / CLASE DE OBJETOS	Personas	Coches	Motocicletas	Camiones
0	0.88305	0.90499	0.75155	0.70093
-20	0.82484	0.90409	0.60544	0.64472
-40	0.46306	0.90477	0.52566	0.69979

Tabla 4.5: Precisiones medias de la evaluación del grupo de imágenes 2, en función del pitch con el que está tomado la imagen de la que se evalúa su detección y la clase de objeto de la detección evaluada.

PITCH / CLASE DE OBJETOS	Personas	Coches	Bicicletas
0	0.87209	0.90094	0.76639
-20	0.74354	0.86356	0.75853
-40	0.83289	0.83303	0.56899

Tabla 4.6: Precisiones medias de la evaluación del grupo de imágenes 3, en función del pitch con el que está tomado la imagen de la que se evalúa su detección y la clase de objeto de la detección evaluada.

PITCH / CLASE DE OBJETOS	Personas	Coches
0	0.88079	0.87486
-20	0.81847	0.85901
-40	0.55876	0.83006

Tabla 4.7: Precisiones medias de la evaluación del grupo de imágenes 4, en función del pitch con el que está tomado la imagen de la que se evalúa su detección y la clase de objeto de la detección evaluada.

Como se explicó en la Sección 4.2.1, un modelo ideal tendría tanto una alta precisión como una alta sensibilidad, esto es, la curva de Precision – Recall se acerca lo más posible a la esquina superior derecha de la gráfica. En nuestros modelos, se puede observar que cuanto más bajo es el valor de pitch (ángulo vertical hacia abajo), peor es la detección. Esto también influye en los valores de la precisión media (Average Precision), los cuales son más bajos en general, cuanto menor sea el valor de pitch. La razón de esto es que cuanto menor sea el valor de pitch, más se deforman los objetos de la imagen. Esto hace que el detector no los detecte con tanta eficacia.

Estas conclusiones no son definitivas puesto que la evaluación se ha realizado sobre un bajo volumen de datos.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Este trabajo de fin de grado tenía como objetivos la creación de una base de datos propia a partir de imágenes que puedan representar todo el espacio a lo largo de rutas, obtenidas de la base de datos de imágenes de Google Street View, así como la prueba de un detector de objetos sobre ellas.

Para alcanzar estos objetivos se realizó un estudio del estado del arte, capítulo 2, en el que se analizan por un lado las herramientas existentes para la extracción de imágenes de Google Street View, Sección 2.1, y en el que también se estudiaron los detectores más comúnmente utilizados para la detección de objetos en imágenes, Sección 2.2. A partir de estas investigaciones se implementó una herramienta capaz de extraer imágenes no panorámicas, según unos parámetros que permiten la representación de todo su entorno. Esta herramienta se desarrolla en la Sección 3.1. Posteriormente se creó un sistema para la prueba de un detector YOLOv3 pre-entrenado con COCO sobre estas imágenes, desarrollado en la Sección 3.2. Por último, se evaluaron en la Sección 4.2.1 los resultados de la detección sobre las imágenes de nuestra base de datos.

Una vez evaluados los resultados, se ha observado que según la imagen extraída de Google Street View y de sus valores de fov, heading y pitch, algunos objetos representados pueden aparecer incompletos, cortados o deformados. A la hora de detectar objetos sobre estas imágenes, el que los objetos aparezcan incompletos o deformados puede empeorar la calidad de la detección en algunos casos.

Así, se puede concluir que los objetivos propuestos en este trabajo, la creación de nuestra propia base de datos con imágenes y la prueba y evaluación de un detector de objetos sobre ella, han sido alcanzados.

5.2 Trabajo futuro

A partir de las conclusiones expuestas, como continuación de este trabajo, en primer lugar, se propone la mejora de la herramienta utilizada para la extracción de imágenes de Google Street View, desarrollada en la Sección 3.1. Esta mejora consistiría en poder extraer las imágenes panorámicas alojadas en Google Street View para poder representar el entorno completo de un punto en una ruta, en una sola imagen. Con ello, una ruta podría representarse más fielmente y con una significativa menor cantidad de imágenes. Así, se podría aumentar la cantidad de rutas extraídas para nuestra base de datos y, con ello, extender el estudio de las detecciones de objetos sobre rutas más variadas.

En segundo lugar, se propone el estudio de la deformación geométrica de los objetos en función de su posición en imágenes no panorámicas de Google Street View, según el ángulo de toma de la imagen y el proceso de formación de la imagen panorámica.

Por último, se propone la prueba de otros tipos de detectores de objetos sobre las imágenes de nuestra base de datos, con el fin de desarrollar un detector más eficiente sobre imágenes con objetos deformados geométricamente. Además, se sugiere aumentar el número de clases de objetos urbanos a detectar.

Referencias

[1] Mirowski, P., Matthew, K. G., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., . . . Hadsell, R. (n.d.). *Learning to Navigate in Cities Without a Map.* Retrieved from https://arxiv.org/abs/1804.00168

[2] Cruz, A. L. (2018, May 23). *Lo más lejos a lo que hemos llegado con la Visión Artificial, ¿cuál es el nuevo reto?* Retrieved from VIU: https://www.universidadviu.es/lo-mas-lejos-lo-que-hemos-llegado-con-la-vision-artificial-cual-es-el-nuevo-reto/

[3] Pezoa, W. L. (n.d.). *Visión artificial y redes neuronales para el reconocimiento facial*. Retrieved from Electro Industria: http://www.emb.cl/electroindustria/articulo.mvc?xid=3179&edi=158&xit=enfoque-combinado-vision-artificial-y-redes-neuronales-para-el-reconocimiento-facial

[4] *Fuentes de las fotografías.* (n.d.). Retrieved from Google Maps: https://www.google.com/intl/es/streetview/explore/

[5] *Developer Guide*. (s.f.). Retrieved from Street View Static API: https://developers.google.com/maps/documentation/streetview/intro

[6] *Google Cloud Platform*. (s.f.). Retrieved from https://cloud.google.com/maps-platform/pricing/?hl=es

[7] *Developer Guide*. (s.f.). Retrieved from Directions API: https://developers.google.com/maps/documentation/directions/intro

[8] *Pano Fetch*. (s.f.). Retrieved from Google Chrome Store: https://chrome.google.com/webstore/detail/pano-fetch/ggmfokbjchlhboclfngkneflhkopebbh

[9] *Street View Download 360*. (s.f.). Retrieved from https://svd360.istreetview.com/

[10] (s.f.). Retrieved from https://mapio.app/

[11] *PanomNom.js*. (s.f.). Retrieved from GitHub: https://github.com/spite/PanomNom.js

[12] *google-streetview 1.2.9*. (s.f.). Retrieved from pipy: https://pypi.org/project/google-streetview/

[13] *Google Maps Streetview Player*. (n.d.). Retrieved from http://www.brianfolts.com/driver/

[14] Bafolts. (n.d.). *google-streetview-images*. Retrieved from GitHub: https://github.com/bafolts/google-streetview-images

[15] Gandhi, R. (2018, July 18). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO—Object Detection Algorithms.* Retrieved from Medium: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[16] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (n.d.). *Rich feature hierarchies for accurate object detection and semantic segmentation.* Retrieved from https://arxiv.org/pdf/1311.2524.pdf

[17] Girshick, R. (n.d.). *Fast R-CNN.* Retrieved from https://arxiv.org/pdf/1504.08083.pdf

[18] Ren, S., He, K., Girshick, R., & Sun, J. (n.d.). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* Retrieved from https://arxiv.org/pdf/1506.01497.pdf

[19] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (n.d.). *You Only Look Once: Unified, Real-Time Object Detection.* Retrieved from https://arxiv.org/pdf/1506.02640v5.pdf

[20] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, May 9). *You Only Look Once: Unified, Real-Time Object Detection.* Retrieved from https://arxiv.org/pdf/1506.02640.pdf

[21] Redmon, J., & Farhadi, A. (n.d.). *YOLO9000: Better, Faster, Stronger*. Retrieved from https://arxiv.org/pdf/1612.08242.pdf

[22] Redmon, J., & Farhadi, A. (n.d.). *YOLOv3: An Incremental Improvement.* Retrieved from https://pjreddie.com/media/files/papers/YOLOv3.pdf

[23] Ouaknine, A. (2018, February 5). *Review of Deep Learning Algorithms for Object Detection*. Retrieved from Medium: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

[24] *The PASCAL Visual Object Classes Homepage*. (n.d.). Retrieved from http://host.robots.ox.ac.uk/pascal/VOC/

[25] *Image-Net*. (n.d.). Retrieved from http://www.image-net.org/

[26] *What is WordNet?* (n.d.). Retrieved from https://wordnet.princeton.edu/

[27] *COCO*. (n.d.). Retrieved from http://cocodataset.org/#home

[28] Hare, J. (n.d.). *StreetviewExtractor.* Retrieved from GitHub: https://github.com/jonhare/StreetviewExtractor

[29] *YOLO: Real-Time Object Detection.* (n.d.). Retrieved from https://pjreddie.com/darknet/yolo/

[30] *evaluateDetectionPrecision.* (n.d.). Retrieved from MathWorks: https://es.mathworks.com/help/vision/ref/evaluatedetectionprecision.html

[31] Ramírez, J. (2018, July 19). *Curvas PR y ROC.* Retrieved from Medium: https://medium.com/bluekiri/curvas-pr-y-roc-1489fbd9a527

[32] *Colisiones Simples AABB.* (2010, May 12). Retrieved from FreakTeam: https://freakteam.wordpress.com/2010/05/12/colisiones-simples-aabb/

Glosario

API	Application Programming Interface
VOC	Visual Object Classification
COCO	Common Objects in Context

Anexos

A Manual StreetView-Stractor

Código original [20].

El siguiente manual tiene como objetivo explicar cómo compilar el proyecto y ejecutarlo.

Entorno de desarrollo

Para trabajar con el proyecto se necesita tener instalados los siguientes programas:

- Java JDK 7
- Apache Maven

Compilación del programa

Desde un terminal, ejecute '*mvn package shade:* shade' para crear un archivo ejecutable.

Ejecución del programa

Se necesita una clave de API de Google
Se debe habilitar la "API de indicaciones" y la "API de imagen de Street View".

Para ejecutar el programa se utiliza la siguiente línea de ejecución:
java -jar target / StreetviewExtractor-1.0-SNAPSHOT.jar [parámetros …]

Parámetros obligatorios:

- Origen: *<línea de ejecución> --from VAL*
- Destino: *<línea de ejecución> --to VAL*
- Api Key: *<línea de ejecución> --api-key (-a) VAL*
- Salida archivo JSON: *<línea de ejecución> --output (-o) VAL*

Parámetros opcionales:

- Tamaño de la imagen: *<línea de ejecución> --width (-w) VAL, <línea de ejecución> --height (-h) VAL*
- Salida de imágenes: *<línea de ejecución> --write-images (-i)*

- Salida de video: *<línea de ejecución> --write-video (-v)*
- FPX: *<línea de ejecución> --fpx VAL*
- TIME-RECODE: *<línea de ejecución> --time-recode*
- HEADING: *<línea de ejecución> --heading VAL*
- HEAD: *<línea de ejecución> --head*
- FOV: *<línea de ejecución> --fov VAL*
- PITCH*: <línea de ejecución> --pitch VAL*
- TRAVEL MODE: *<línea de ejecución> --mode VAL*

Siento VAL el valor elegido para cada parámetro.

Ejemplo de línea de ejecución:

java -jar target/StreetviewExtractor-1.0-SNAPSHOT.jar --from 41.388520,2.171659 --to 41.385920,2.170060 -i -o test.json -a --head --fov 100 –pitch 10 --mode WALKING

Archivos de salida

Las salidas, que se colocarán en el archivo "target" son:

- Archivo Geojson con información de la ruta.
- Archivo con las imágenes a lo largo de la ruta llamada: *<OutputValue>_<ModeRoute>_+<HeadingValue>_<PitchValue>_<FovValue>-jpegs*

  Las imágenes dentro de este archivo se denominan:

  <num_image>_<longitude>_<latitude>_<HeadingValue>_<PitchValue>_<FovValue>.jpeg

  Siendo *num_image* el número (en 3 cifras) de la imagen en la ruta.